



**SYSTEM PROVISIONING AND CONFIGURATION
MANAGEMENT LAB**

**Lab File
(2023-2024)**

for

6th Semester

Submitted To

Dr. Hitesh Kumar Sharma
Cluster Head (Cybernetics)
School of Computer Science

Submitted By:

Arpit Goyal
B. Tech. CSE DevOps [6th
Semester]
500094790
R2142210148
B-3

Exercise 3–Provisioning an EC2 Instance on AWS

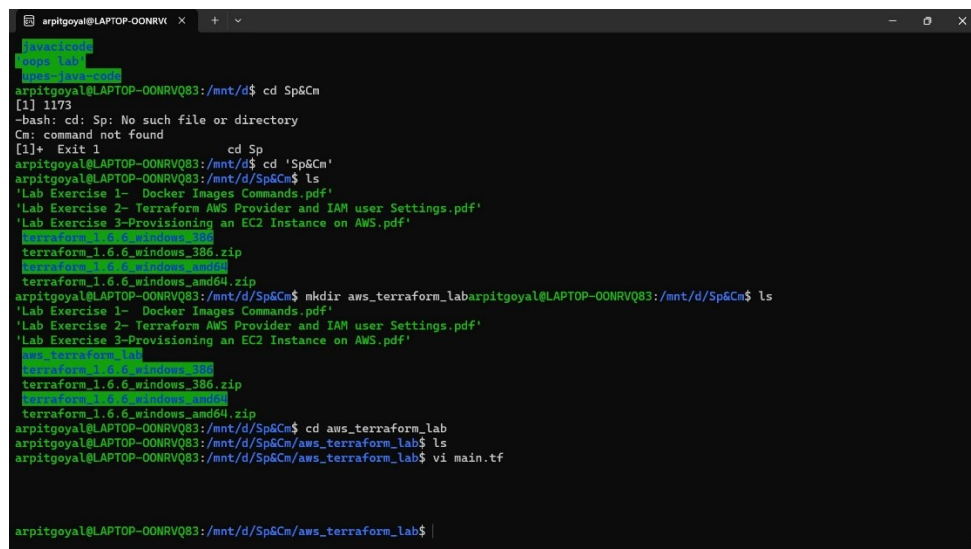
Prerequisites: Terraform Installed: Make sure you have Terraform installed on your machine.

AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

Exercise Steps:

Step 1: Create a New Directory:

Create a new directory for your Terraform configuration:

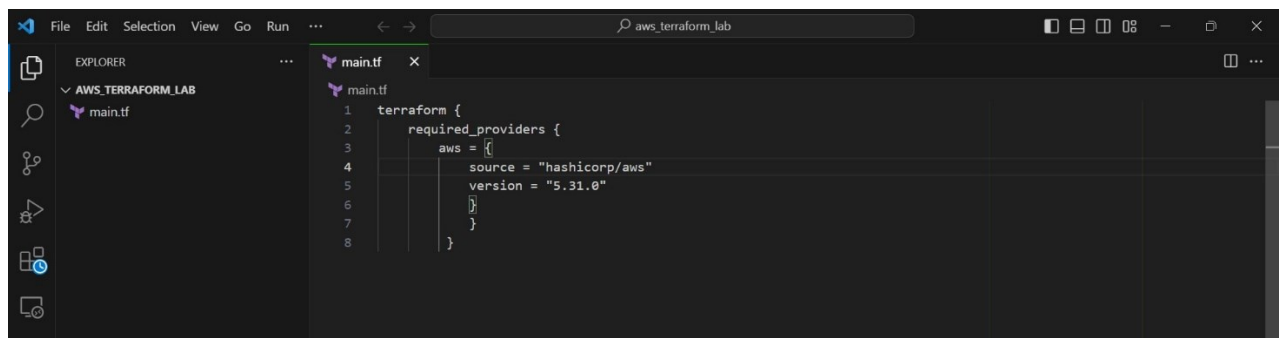
A terminal window showing a series of commands and their outputs. The user navigates to a directory, creates a subdirectory named 'aws_terraform_lab', and lists the contents of the parent directory. The output shows several PDF files and zip files related to Terraform and AWS.

```
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d$ cd Sp&Cn
[1] 1173
-bash: cd: Sp: No such file or directory
Cm: command not found
[1]+  Exit 1                  cd Sp
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d$ cd 'Sp&Cn'
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn$ ls
'Lab Exercise 1- Docker Images Commands.pdf'
'Lab Exercise 2- Terraform AWS Provider and IAM user Settings.pdf'
'Lab Exercise 3-Provisioning an EC2 Instance on AWS.pdf'
aws_terraform_lab
terraform-1.6.6-windows_386.zip
terraform-1.6.6-windows_amd64.zip
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn$ mkdir aws_terraform_lab
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn$ ls
'Lab Exercise 1- Docker Images Commands.pdf'
'Lab Exercise 2- Terraform AWS Provider and IAM user Settings.pdf'
'Lab Exercise 3-Provisioning an EC2 Instance on AWS.pdf'
aws_terraform_lab
terraform-1.6.6-windows_386.zip
terraform-1.6.6-windows_amd64.zip
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn$ cd aws_terraform_lab
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn/aws_terraform_lab$ ls
arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn/aws_terraform_lab$ vi main.tf

arpitgoyal@LAPTOP-00NRVQ83: /mnt/d/Sp&Cn/aws_terraform_lab$
```

Step 2: Create Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

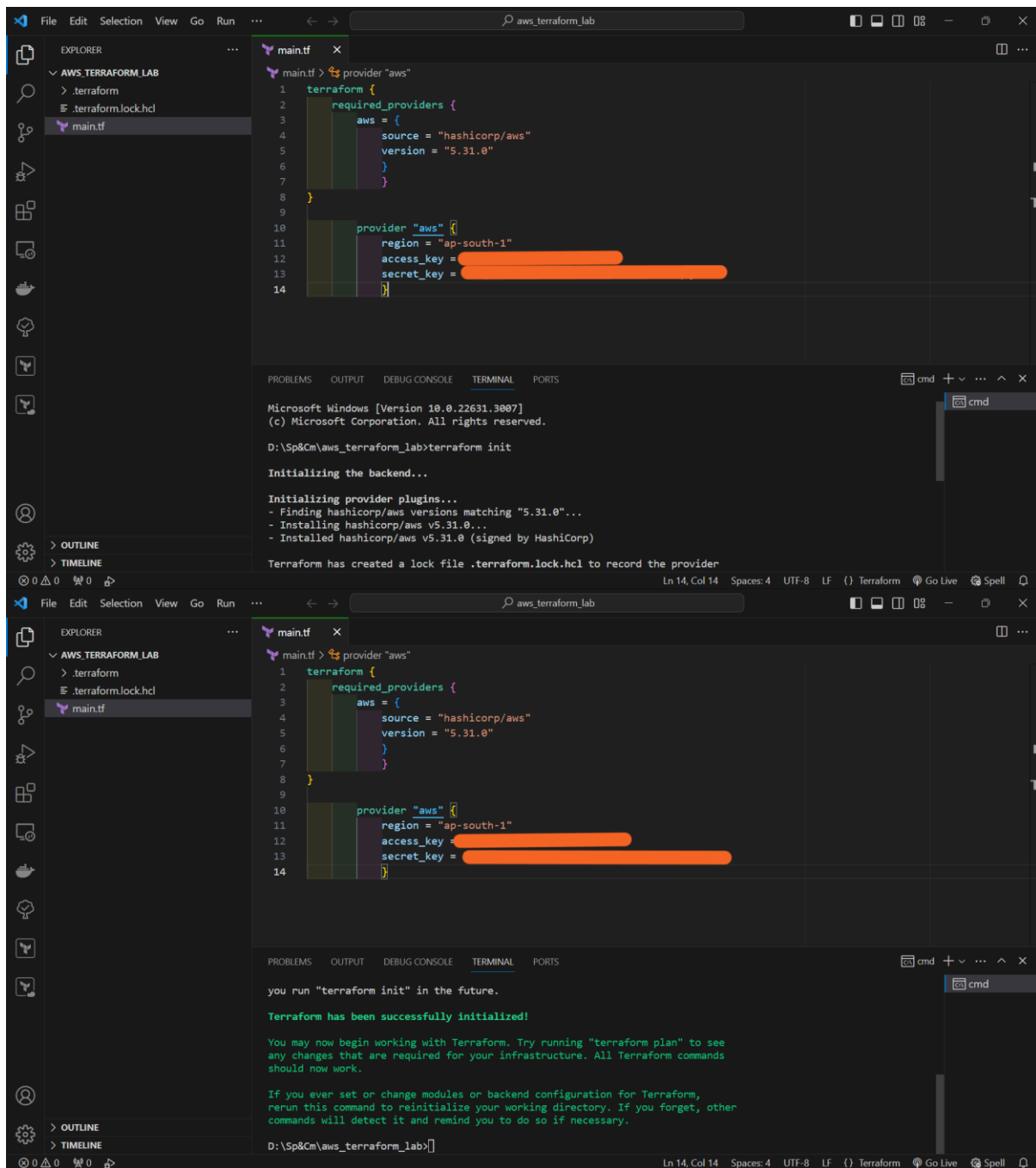
A screenshot of a code editor (VS Code) showing the content of a file named 'main.tf'. The file contains Terraform configuration for the AWS provider.

```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.31.0"
6     }
7   }
8 }
```

This script defines an AWS provider and provisions an EC2 instance.

Step 3: Initialize Terraform:

Run the following command to initialize your Terraform working directory:



```
main.tf > provider "aws"
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.31.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "ap-south-1"
12   access_key = 
13   secret_key = 
14 }
```

```
Microsoft Windows [Version 10.0.22631.3007]
(c) Microsoft Corporation. All rights reserved.

D:\Sp&Cm\aws_terraform_lab>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
```

```
main.tf > provider "aws"
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.31.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "ap-south-1"
12   access_key = 
13   secret_key = 
14 }
```

```
you run "terraform init" in the future.

Terraform has been successfully initialized!

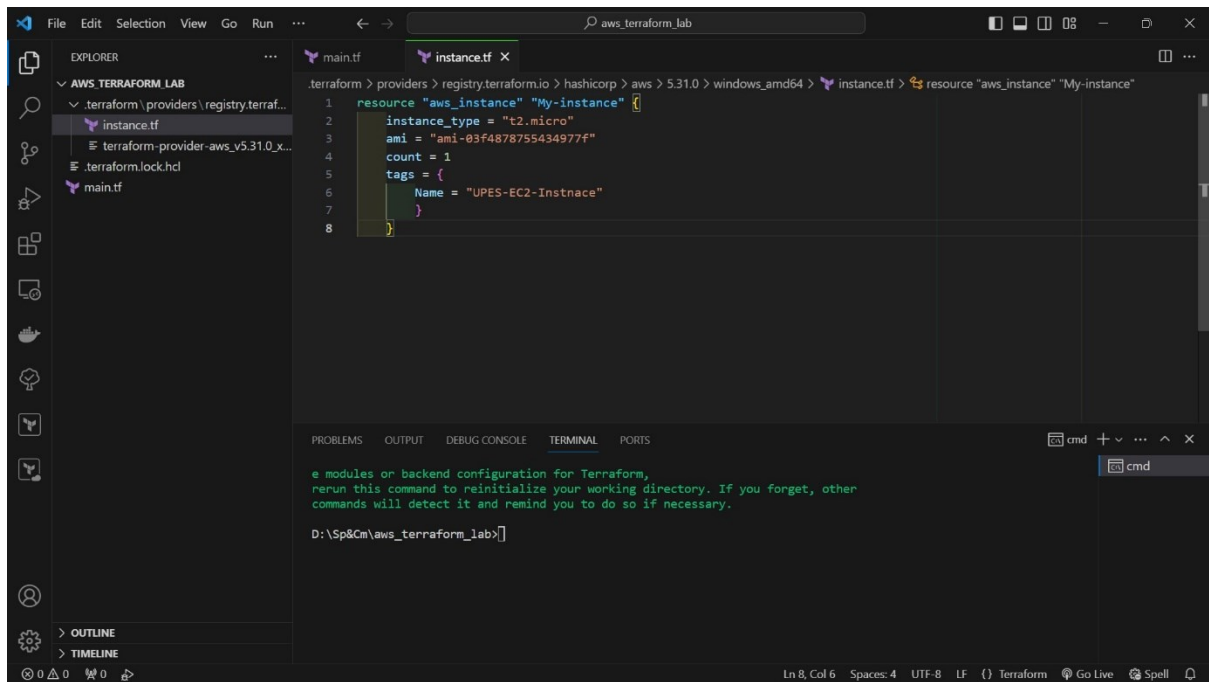
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

D:\Sp&Cm\aws_terraform_lab>
```

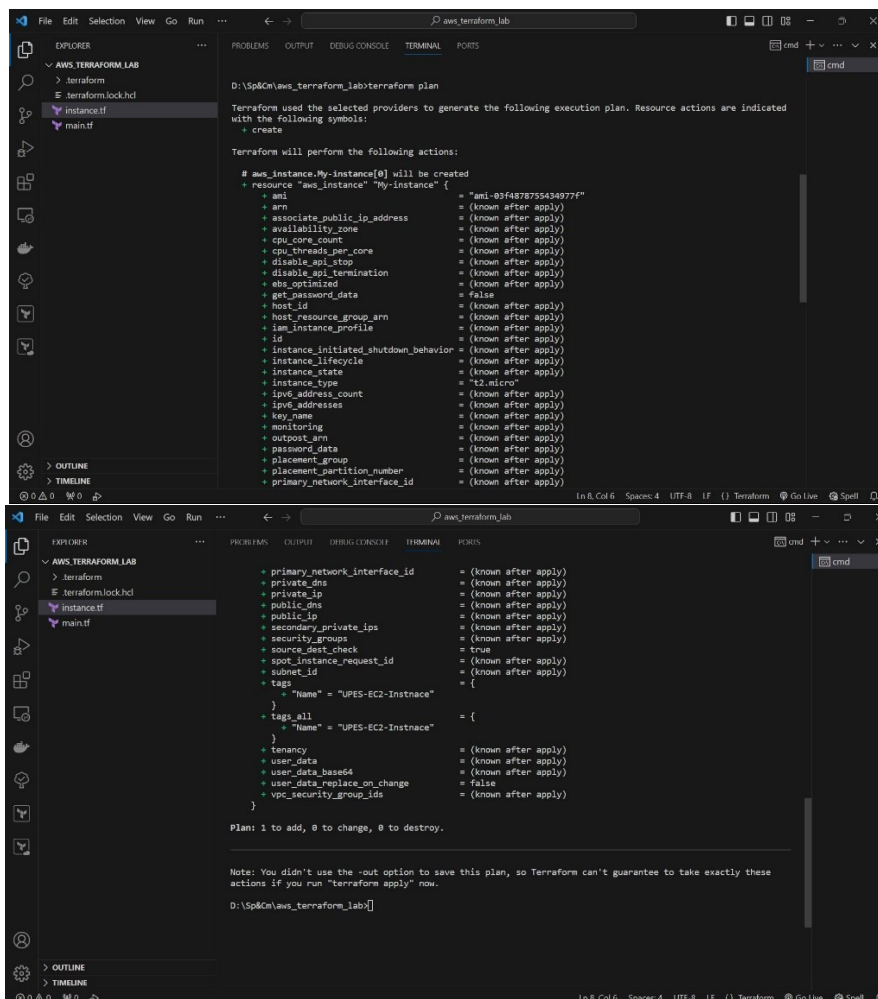
Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):

Create a file named instnace.tf with the following content:



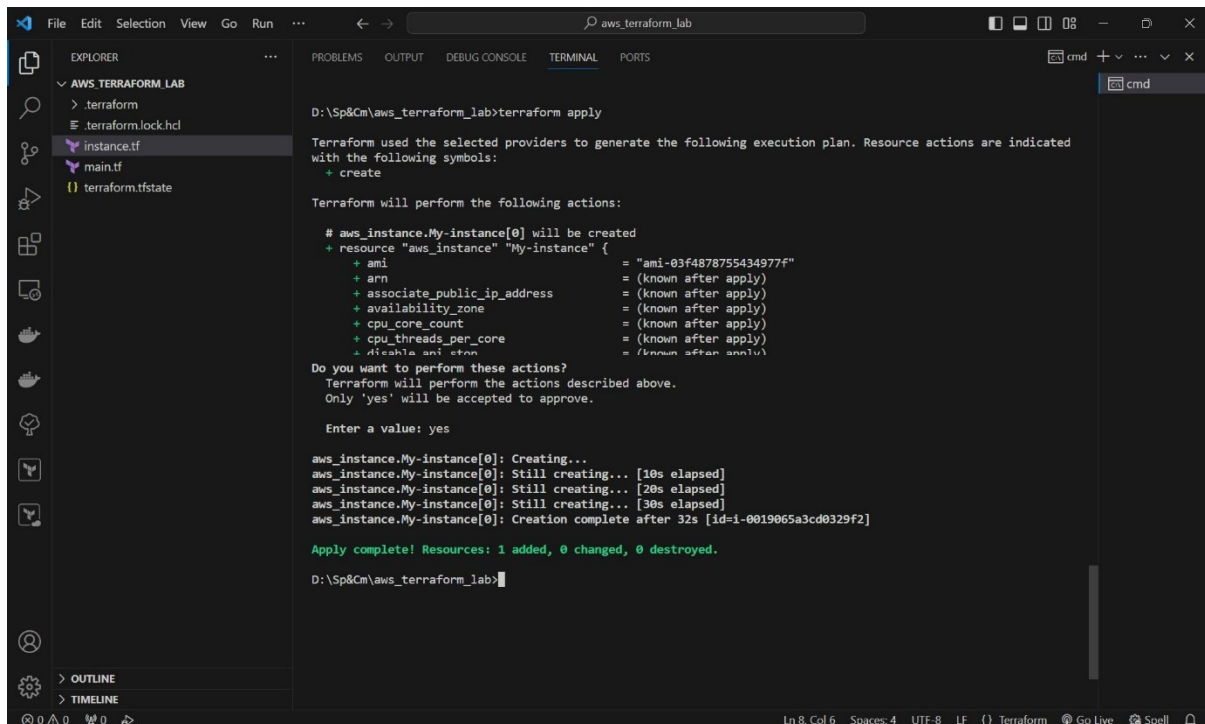
Step 5: Validate and Review Plan:

Run the following command to see what Terraform will do: `terraform validate` and `terraform plan`



Step 6: Apply Changes:

Apply the changes to create the AWS resources:
Type yes when prompted.



```
D:\Sp8Cm\aws_terraform_lab>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.My-instance[0] will be created
+ resource "aws_instance" "My-instance" {
  + ami               = "ami-03f4878755434977f"
  + arn               = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone  = (known after apply)
  + cpu_core_count    = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized      = (known after apply)
  + enable_monitoring  = (known after apply)
  + hibernation        = (known after apply)
  + iam_instance_profile = (known after apply)
  + instance_type      = (known after apply)
  + key_name            = (known after apply)
  + monitoring          = (known after apply)
  + network_interface_id = (known after apply)
  + placement_group     = (known after apply)
  + primary_monitoring_enabled = (known after apply)
  + root_block_device   = (known after apply)
  + security_groups     = (known after apply)
  + source_dest_check    = (known after apply)
  + subnet_id           = (known after apply)
  + tags                = (known after apply)
  + user_data            = (known after apply)
  + vpc_security_group_ids = (known after apply)
}

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

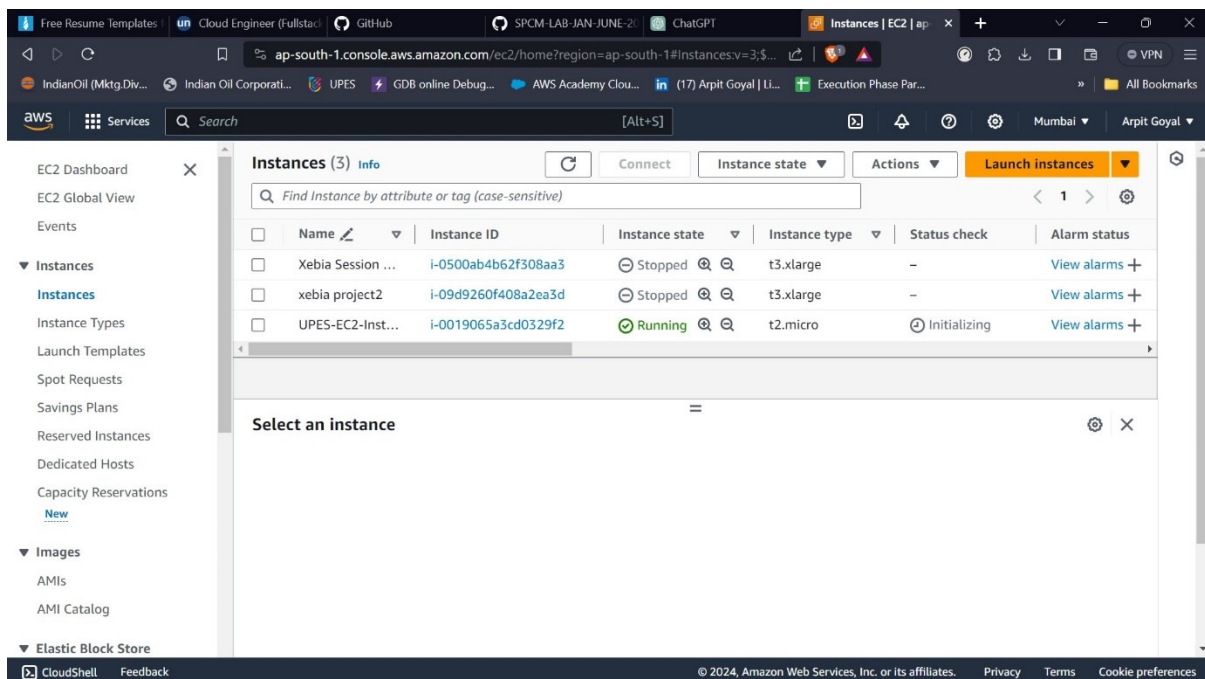
aws_instance.My-instance[0]: Creating...
aws_instance.My-instance[0]: Still creating... [10s elapsed]
aws_instance.My-instance[0]: Still creating... [20s elapsed]
aws_instance.My-instance[0]: Still creating... [30s elapsed]
aws_instance.My-instance[0]: Creation complete after 32s [id=i-0019065a3cd0329f2]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

D:\Sp8Cm\aws_terraform_lab>
```

Step 7: Verify Resources:

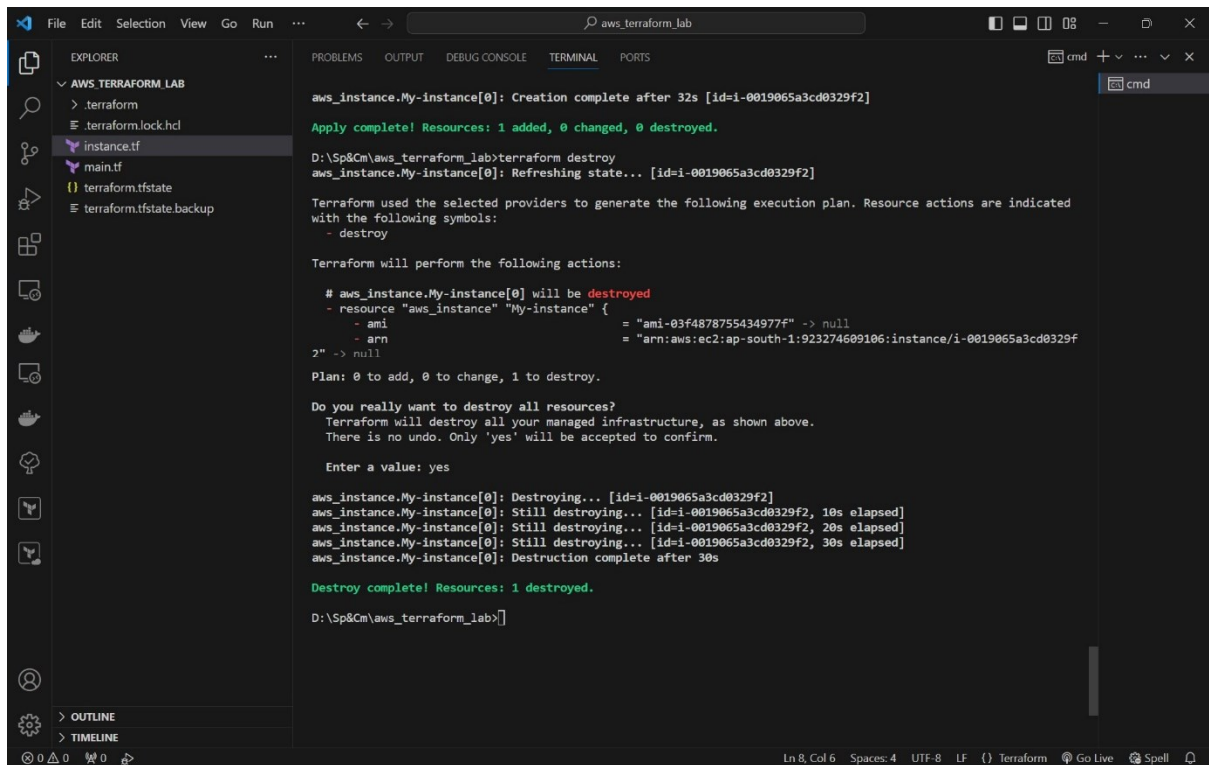
After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.



Step 8: Cleanup Resources:

When you are done experimenting, run the following command to destroy the created resources:

Type yes when prompted.



The screenshot shows a VS Code editor with a terminal window open. The terminal displays the output of the Terraform destroy command. It shows the plan for destroying the aws_instance.My-instance[0] resource. The plan indicates that the resource will be destroyed. The user is prompted to confirm the destruction, and they enter 'yes'. The terminal then shows the destruction process, including the time taken to destroy the resource (30s).

```
aws_instance.My-instance[0]: Creation complete after 32s [id=i-0019065a3cd0329f2]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
D:\Sp&Cm\aws_terraform_lab>terraform destroy
aws_instance.My-instance[0]: Refreshing state... [id=i-0019065a3cd0329f2]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.My-instance[0] will be destroyed
- resource "aws_instance" "My-instance" {
  - ami           = "ami-03f48755434977f" -> null
  - arn           = "arn:aws:ec2:ap-south-1:923274609106:instance/i-0019065a3cd0329f2" -> null
  - instance_type = "t2.micro" -> null
}

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.My-instance[0]: Destroying... [id=i-0019065a3cd0329f2]
aws_instance.My-instance[0]: Still destroying... [id=i-0019065a3cd0329f2, 10s elapsed]
aws_instance.My-instance[0]: Still destroying... [id=i-0019065a3cd0329f2, 20s elapsed]
aws_instance.My-instance[0]: Still destroying... [id=i-0019065a3cd0329f2, 30s elapsed]
aws_instance.My-instance[0]: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.

D:\Sp&Cm\aws_terraform_lab>
```

